REPORT:

Deep Q learning to solve a navigation problem

Description

As described in README.md, in this project we solve a navigation problem where an agent has to explore a large square world and try to get as many yellow bananas as possible while avoiding the blue bananas.

In order to solve this problem, we have selected the well known Deep Q learning approach. It is similar to the tabular Q-learning algorithm, but instead of using a lookup table or dictionary to store each state action pair values, it uses function approximator to find the underlying relationships between the input (state) and the output (action). In this case, a non linear function approximator is it used, a neural network.

Reinforcement learning is known to be unstable when used with a nonlinear function approximator such as neural networks. Hence, this instabilities have been addressed with experience replay and target network as explained in Human-level control through deep reinforcement learning. This mechanisms are used to minimized the correlation:

- Between the sampled experiences
- Between the target (calculated in each visit inside the episodes) and the action-value parameters (weights of the neural network) that are constantly updating.

The developed code it is based on the codebase provided in the lessons of the Deep Reinforcement Learning Nanodegree of Udacity.

Learning algorithm

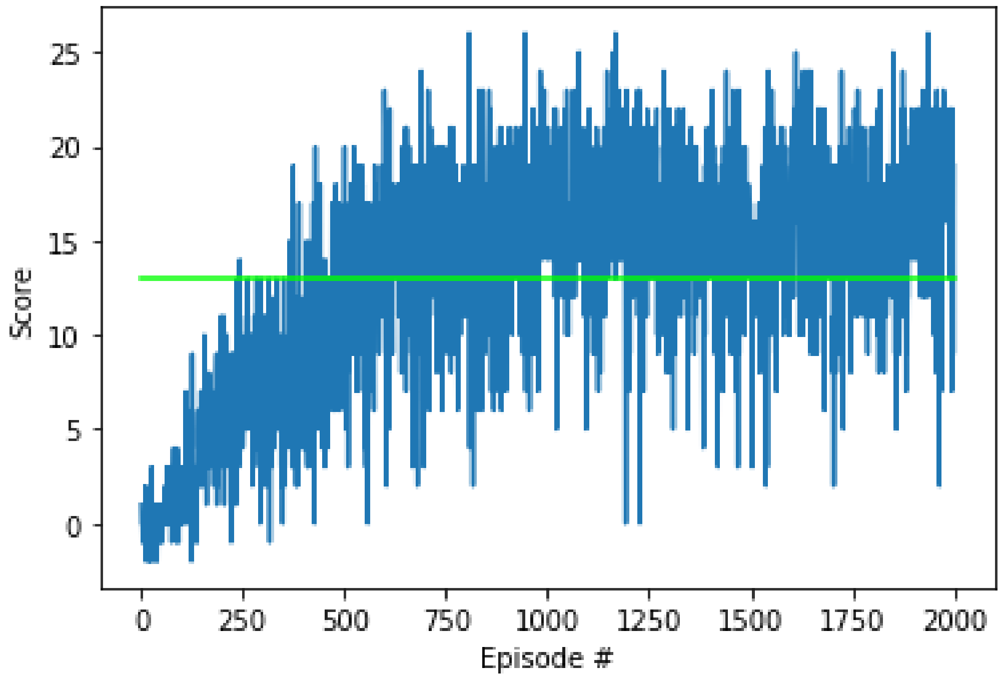
DQN update rule

Hyperparams

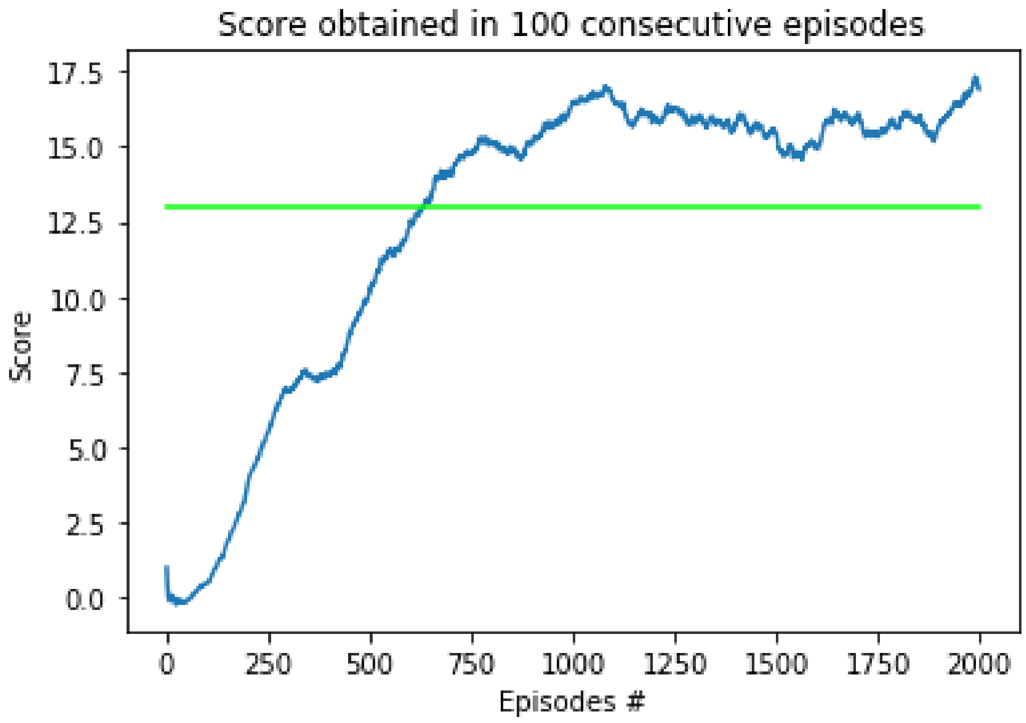
NN architecture

Plot of rewards

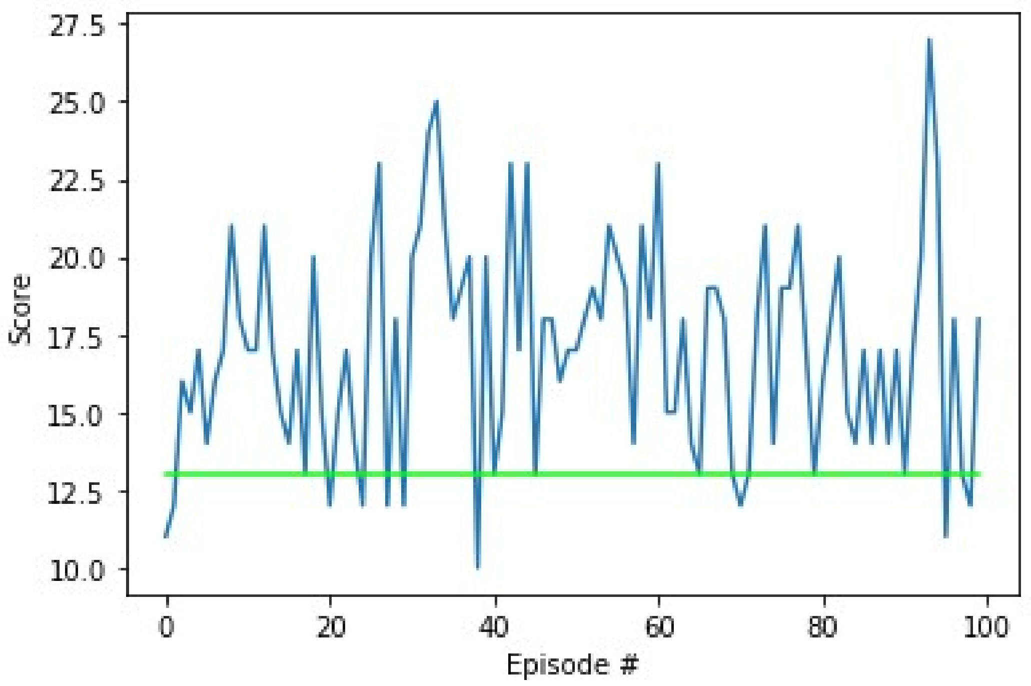
The problem is considered solved when the agent is able to average a score of +13 over 100 consecutive episodes. This is achieved in the training after 631 episodes, although I let the agent train a little bit further to see how keeps learning.



To make it easier to detect when has achieved the goal, I attached another plot when it could be seen the acumulative score of the last 100 episodes. I.e. when X-axis is 500, the projected score is the average score of the episodes between 400 and 500.



Furthermore, I attached below a plot where it could be seen the obtained score when we evaluate the model for 100 episodes. The graph shows that after the training phase, the agent gets an average score of 17.02 in each episode.



Ideas for Future Work

In the future, it would be interesting to implement other well-known approaches such as Dueling Networks and Double DQN to improve the performance of the agent. Moreover, Policy-based methods might work well too, so in the future it would be something interesting to test.